VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza střeleckých terčů z optických obrazů
Detection of Shooting Targets from Images

2016 Richard Schneider

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Richard Schneider**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Analýza střeleckých terčů z optických obrazů
Detection of Shooting Targets from Images

Jazyk vypracování: čeština

Zásady pro vypracování:

S rozvojem kamerových systémů se v posledních letech zvýšilo i jejich využití v oblasti sportovních aktivit; například detekce a sledování pohybu sportovců, detekce brankové čáry, měření vzdálenosti. Další možností využití kamerových systémů se nabízí v oblasti sportovní střelby na detekci a vyhodnocení zásahů v terčích.
Cílem této práce bude vytvoření programu pro analýzu těchto terčů.

1. Seznamte se se základními pojmy v oblasti detekce objektů v obrazech.
2. V popisu se zaměřte zejména na metody, které by mohly být vhodně využity pro analýzu terčů (segmentace obrazu, porovnání vzorů, detekce kružnic).
3. Seznamte se s knihovnou OpenCV.
4. S pomocí knihovny OpenCV tento detektor implementujte.
5. Experimentálně ověřte funkčnost, přesnost a rychlost detektoru.
6. Své závěry řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radovan Fusek**

Datum zadání: 01.09.2014

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem bakalářskou práci na téma Analýza střeleckých terčů z optických obrazů zpracoval sám. Veškeré prameny a zdroje informací, které byly použity k sepsání této práce, byly řádně citovány v poznámkách pod čarou a jsou uvedeny v seznamu použitých pramenů a literatury.

V Těrlicku dne 26.04.2016

Rád bych poděkoval vedoucímu mé bakalářské práce panu Ing. Radovanu Fuskovi za cenné rady při vytváření této práce. Děkuji také všem, kteří mě podporovali po celou dobu studia.

Abstrakt

Cílem této bakalářské práce je prozkoumat metody vhodné pro vyhledání zásahu ve vyfoceném terči. Z prozkoumaných metod vybrat tu, která bude mít nejlepší výsledky, a s danou metodou naimplementovat mobilní aplikaci pro analýzu střeleckých terčů. V první části popíšu obecné principy analýzy, dále se zaměřím na metody vhodné pro analýzu terčů a popíšu jejich nevýhody a výhody. Další část bude zaměřená na základní využití knihovny OpenCV pro operační systém Android. V následující kapitole popíši uživatelské rozhraní aplikace. Poté popíši svůj algoritmus pro získání zásahů z obrazu terče. Předposlední kapitola bude zaměřená na test přesnosti a rychlosti analýzy a dosažené výsledky vyhodnotím. V závěrečné kapitole navrhnu možné budoucí vylepšení aplikace pro analýzu terčů.

Klíčová slova: Analýza, terč, OpenCV, Java, Android, mobilní aplikace, obraz

Abstract

The aim of this thesis is defined useful methods for searching interference in intercepted target. Based on this searching is possible choose one of method with the best output and implement mobile application for the analysis of shooting targets. Theoretical part describes general principle of analysis, methods suitable for shooting targets analysis and specialises in their disadvantages and advantages. Next part contains basic usage library OpenCV for operation system Android. User interface for application is defined as next. This thesis includes algorithm gained interference from image of target. Practical part mentions very important test of analysis focussed on speed and this outputs are evaluated. In the final chapter is designed future possible improvements for shooting targets application.

Key words: Analysis, target, OpenCV, Java, Android, mobile application, image

Obsah

Seznam použitých zkratek a symbolů

OpenCV	-	Open Computer Vision
RAM	-	Random Access Memory
CPU	-	Central Processing Unit
BSD	-	Berkeley Software Distribution

Seznam tabulek

Seznam ilustrací

1 Úvod

Vzhledem k tomu, že téměř každý mobilní telefon má poměrně kvalitní fotoaparát a většina lidí nějaký mobilní telefon vlastní, je vývoj aplikací spočívajících v analýze vyfocených obrazů na vzestupu. Dle statistických údajů Policejního prezídia ČR, ředitelství služby pro zbraně a bezpečnostní materiál, vlastnilo ke dni 31. 12. 2014 jen v České republice střelnou zbraň zhruba 292 000 lidí a v držení osob se nacházelo zhruba 752 000 zbraní[1]. Lze očekávat, že většina z držitelů palných zbraní používá zbraně na trénink na střelnicích střelbou do terče. Následné spočítání a vyhodnocení nastřílených terčů trvá poměrně dlouho a není vždy přesné. Po prozkoumání trhu s mobilními aplikacemi jsem došel k závěru, že aplikace pro výpočet zásahů v terči, pomocí vyfocení terče a následné analýzy, není k dostání na trhu, a z tohoto důvodu jsem se pokusil vytvořit vhodnou aplikaci pro analýzu zásahů v terči. Většina kolegů střelců, kterých jsem se dotázal, by takovouto aplikaci pro běžné použití uvítala. První část bude zaměřena na základní pojmy a metody, které mohou být využity pro práci s obrazem. V této části bude popsáno pět základních fází (snímání a digitalizace, předzpracování, segmentace, popis, klasifikace), které popisují proces zpracovávání obrazu. Poté popíši metody vhodné pro analýzu střelecký terčů z vyfocených obrazů, které vyzkouším a vyhodnotím jejích použití pro mou aplikaci. V následující části uvedu základní informace ke knihovně OpenCV, která nově pracuje i pod operačním systémem Android, a popíši její základní funkce pro práci s obrazy. Pátá kapitola se již zaměří na uživatelské rozhraní, ovládací prvky a posloupnost kroků k dosažení analýzy terčů a možnosti, které uživatel bude mít. Šestá kapitola je popis algoritmu, kterým se docílí, aby zásahy v terči bylo možné analyzovat. Jedná se o popis nezbytných kroků a funkcí v nich použitých. Každý krok analýzy bude ukázán i na obrázku. V sedmé kapitole provedu testy své aplikace. Testy budu provádět na rozdílných mobilních zařízeních, a to s rozdílným typem procesoru a rozdílným rozlišením fotoaparátu. Dále se zaměřím na kvalitu, rychlost a správnost rozeznání zásahů. Výsledky budou zpracovány do přehledných tabulek a poté i vyhodnoceny. V osmé kapitole navrhnu pár možných modifikací a vylepšení, aby aplikace byla přesnější a kvalitnější. Na závěr shrnu výsledky testů a vyhodnotím možnost jejího využití v reálném životě.

[1] LEX. [online]. [cit. 2016-03-18]. Dostupné z: http://gunlex.cz/clanky/informace-lex/2002-pocet-drzitelu-zbrojnich-prukazu-a-zbrani-v-roce-2014

2 Analýza obrazu

Úplně prvním krokem pro analýzu obrazu pomocí počítače je uvědomit si, jakým způsobem počítač obraz vidí. Obraz můžeme popsat matematicky jako spojitou funkci dvou či tří argumentů, které lze vyjádřit vzorcem:

$$f = (x, y) \text{ případně } f = (x, y, z).$$

Ve většině případů je funkce f = (x, y) nazývána jako funkce obrazová. Proměnnými x, y chápeme reálná čísla, která představují souřadnice v rovině. Nejčastěji měřená fyzikální veličina je jas neboli intenzita, ale může se jednat i o jinou veličinu, např. teplotu aj.[2]

Z pohledu počítačového vidění je analýza obrazu proces, který má za úkol najít v daném obraze nějaký určený objekt.

Zpracování obrazu můžeme rozdělit do pěti základních kroků, a to těchto:

- Snímání a digitalizace
- Předzpracování
- Segmentace obrazu
- Popis
- Klasifikace.

2.1 Snímání a digitalizace

Aby bylo možné obraz zpracovat a dále analyzovat, je nutné jej nasnímat. Snímání obrazu je v posloupnosti činností, které musíme učinit k analýze obrazu, prvním krokem, který ovlivní celý proces zpracování obrazu. Pokud zvolíme správné podmínky, jako jsou např. poloha, osvětlení či konkrétní vlastnosti zařízení, kterým obraz zachycujeme, ovlivníme pozitivně další průběh zpracování. Naopak pokud obraz bude ve špatné kvalitě, nemusí být výsledek správný.

2.2 Předzpracování

Úkolem této fáze procesu je buď odstranění nechtěných či rušivých informací, které jsou obsaženy v obraze, např. šumu či jiného zkreslení, které vzniklo digitalizací nebo přenosem obrazu, nebo touto fází zvýrazníme znaky, které považujeme za podstatné pro další zpracování, a to např. převodem do šedotónového obrazu, zvýrazněním zvolených hran atp.

Předzpracování probíhá např. pomocí jasových transformací, kdy je každému pixelu přiřazena nová hodnota jasu, a to dle transformační funkce (např. negativ). Dále může být provedeno předzpracování pomocí geometrické transformace, která slouží k cílenému zkreslení obrazu či opravě daného obrazu, ať už z důvodu nevhodně zvolené polohy při snímání či špatného snímacího zařízení. K základním geometrickým transformacím řadíme posunutí, změnu měřítka nebo otočení.[3] Ještě než přejdeme do další fáze analýzy, je vhodné, a v některých případech i nutné, zbavit obraz aspoň částečně šumu,

[2] M. Šonka, V. Hlaváč: Počítačové vidění, Grada. Praha 1992, ISBN 80–85424–67–3.

[3] M. Dobeš: Zpracování obrazu a algoritmy v C#, BEN – technická literatura. Praha 2008–233–6, ISBN 978–80–7300.

který se do obrazu dostal snímáním nebo při přenosu. Obraz obsahuje osamocené a výrazně odlišené obrazové body od okolí. Vyhlazovací metody se pokouší tyto pixely nahradit pravděpodobnější hodnotou, která je získávána na základě hodnot jasu bodů nacházejících se v okolí. Negativním jevem vyhlazování je ve většině případů rozmazávání hran, které mají v obraze taktéž výrazné změny jasu. Pokud se snažíme naopak hrany zvýraznit ostřením, zvýšíme i šum.

2.3 Segmentace

Segmentace obrazu na objekty je snaha najít v obraze objekty, které nás zajímají a rozlišit je od pozadí. Zjednodušeně můžeme říct, že se jedná o rozdělení nějakého obrazu na menší části. Každá z části obrazu by po dokonalé segmentaci měla být jedním z objektů obrazu. Rozlišujeme segmentaci kompletní (byly nalezeny oblasti, které odpovídají objektům v reálném světě a zároveň se nepřekrývají) nebo částečnou (v případech, kdy objekty přesně nesouhlasí s objekty). Dalším přínosem segmentace je podstatné omezení objemu dat, což urychluje další zpracování. U segmentace je důležitým hlediskem výběr správného algoritmu, který bude jiný, pokud budeme analyzovat fotografie terčů nebo obrazy z magnetické rezonance. Rozlišujeme několik metod segmentace, a to prahování, spojování oblastí, dělení oblastí, detekce hran či srovnání se vzorem. U prahování se pomocí hodnot jasu obrazového bodu a správně zvoleného prahu určí příslušnost obrazového bodu buď k popředí, nebo k pozadí. Po prahování získáme dvouúrovňový obraz. U metody spojování oblastí je na začátku obraz rozdělen na několik malých částí (klidně i na úrovni pixelů), které následně spojujeme na základě daných podmínek (např. informace o barvě oblasti), a to tak dlouho, dokud již není možné v obraze žádnou další oblast spojit. Opačný postup než u spojování oblastí je u metody dělení oblastí. Celý obraz, který je na začátku v jedné oblasti, dělíme podle určených vlastností na menší. Další z metod je detekce hran. Hranou se rozumí výrazná změna jasu. U této metody se vyhledávají tyto velké změny v jasu. Na pohled jednoduchou metodou je srovnávání se vzorem, ovšem není to zpravidla pravda. V novém obraze je na rozdíl od původního obsažen šum, neshoduje se přesně s původním objektem např. díky geometrické deformaci, pootočení. Z tohoto důvodu není tedy vyhledávána přesná kopie vzoru, nýbrž dostatečná podobnost, která je vyjádřená korelačním koeficientem v hodnotách $\langle -1, 1 \rangle$.

2.4 Popis

U popisu můžeme používat kvalitativní nebo kvantitativní přístup. U prvního z nich jsou popisovány vztahy mezi objekty a vlastnosti, které mají. Vztahy můžeme vyjádřit grafy typu stromu či řetězci. U kvantitativního ukládáme objekty pomocí číselných charakteristik. Vlastnosti, které takto ukládáme, mohou být např. velikost, kompaktnost.

2.5 Klasifikace

Jedná se o poslední část procesu analýzy obrazu. V této fázi by měl být rozpoznán objekt, tzn. že by měl být obraz přiřazen do jednotlivých tříd. Klasifikace je úzce propojena s popisem objektů v obrazech. Klasifikátorem se rozumí algoritmus, který rozhoduje o příslušnosti objektu k dané třídě, a to na základě popisu daného objektu.

Ilustrace 1: schematické znázornění klasifikace

Rozlišujeme rozpoznávání příznakové nebo syntaktické, a to dle způsobu popisu tvořícího vstup do klasifikátoru. Důležitou roli zastává při klasifikace kromě správně zvolených charakteristik také nastavení klasifikátoru. K nastavení klasifikátoru může být použita trénovací množina, tj. soubor prvků, u kterých je předem známa příslušnost k třídě. Nemůžeme-li využít trénovací množinu, protože nevíme počet tříd, nebo ji nemáme k dispozici, využijeme metodu shlukové analýzy. Při této metodě jsou objekty rozděleny dle rozpoznaných objektů do shluku.

3 Metody rozpoznání objektů v obraze

Metod pro rozpoznání objektů v obraze je spousta, některé fungují pomocí rozpoznaní určité barvy, jiné na rozpoznání tvaru, či rozdílného jasu v obraze. Pro analýzu terčů a rozpoznání zásahu v nich jsem nenašel žádnou doporučenou metodu. Terč bude uživatelem snímán pomocí kamery v mobilním zařízení, i když tyto kamery na svou velikost mají dobré parametry, kvalita výsledného obrazu není vždy dostačující pro každou rozpoznávací metodu. Dalším limitujícím faktorem je osvětlení. Obraz terče nebude vždy stejně osvětlený, a proto je zapotřebí použít takovou metodu, která nebude závislá na jasu. Jednou z použitelných metod je metoda nalezení vzoru v obraze neboli Template Matching

3.1 Template Matching

Template Matching je metoda vyhledávající vzory v obraze. Knihovna OpenCV v dnešní době nabízí celkem šest metod pro hledání vzorů v obraze. První moje pokusy byly s metodu CV_TM_SQDIFF, která dle dokumentace OpenCV je dostatečně přesná a přesto nevyžaduje velký výpočetní výkon.

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

Kde *I* je prohledávaný obraz, *T* hledaný vzor, *x, y* jsou souřadnice současné pozice v obraze a *x', y'* jsou souřadnice vzoru. K metodě je potřeba obraz terče a šablona, která v mém případě bude odpovídat vystřiženému zásahu. Šablona bude po jednotlivých pixelech posunována po obraze a pomocí vzorce výše bude vypočítána shoda s obrazem terče. Výstupní obraz je ve stupních šedi, a tam kde je shoda šablony s obrazem menší, tam bude bod tmavší. Tyto body lze v obraze jednoduše najít pomocí funkce cvMinMaxLoc, která je v OpenCV již naimplementovaná.

Analýza terče pomocí použití metody Template Matching trvala na mém mobilním zařízení dlouhou dobu, a proto jsem se pokusil jí zrychlit pomocí snížení rozlišení obrazu. To jsem uskutečnil pomocí rozmazávací funkce Blur, kterou popíši v další kapitole. Bohužel ani tak se doba analýzy výrazně nezkrátila, protože i funkce Blur tvá na mobilním zařízení podstatně dlouhou dobu a výsledný obraz není tak kvalitní jako po použití funkce Gaussian Blur, která ale trvá delší dobu.

Další nevýhodou v hledání vzoru při analýze terčů je, že každý zásah nevypadá stejně. Zásah může mít jinak roztržený okraj po průniku projektilem, jeho vnitřek může mít jinou barvu v důsledku jiného použitého podkladu pod terč, zásah v terči vypadá jinak za šera, nebo naopak když je terč osvícen přímým sluncem. Zásah také může být umístěn na bodovém kruhu na terči a tudíž jeho okraj má jinou barvu. Tato metoda si také neumí poradit se soustřely, kdy je více ran v jednom místě.

S touto metodou jsem nedosahoval dobrých výsledků, co se týče přesnosti a kvality. Tato metoda hlavně potřebuje velký výpočetní výkon, který v mobilních zařízeních není. Dalším krokem bylo použít metodu na filtraci určité barvy.

3.2 Nalezení objektu pomocí určité barvy

Při této metodě jsem použil funkci inRange, kterou popisuji v další kapitole. Oproti metodě Temple Matching není metoda na filtraci barvy závislá na tvaru zásahu. Pomocí této metody, se nalezne v obraze barva, která bude odpovídat zásahu a ten se zvýrazní bílým bodem na černém plátně. Analýza pomocí filtrace barvy je velice přesná, ale uživatel musí pod terč dát podložku dané barvy, nejlépe takové, která se neobjevuje nikde na terči a je dost výrazná.

Při testování jsem pod terč dával červenou podložku. Červená barva byla dost kontrastující a šla lehce vyhledat a zásahy byly dobře čitelné. Rychlost analýzy byla vysoká z důvodu menší náročnosti na předzpracování obrazu. Výsledky za použití této metody jsou velice dobré a bylo možné vyhodnotit i soustřely.

Ale ani tuto metodu jsem do své aplikace nepoužil. Nutnost podkládat terč červenou podložkou je uživatelsky limitující. Sice bylo možné podložku použít, jak před prostřelením terče, tak až poté, a barva podložky se dala nastavit, ale ne vždy má uživatel možnost takto si terč podložit. Proto jsem následně zkusil metodu na detekci hran.

3.3 Detekce hran

Detekce hran patří mezi metody založené na hranici. Hranice jsou tvořené hranami, které lze v obraze najit za pomocí funkce Canny, popis v další kapitole. Před hledáním hran je třeba obraz upravit, aby nedocházelo k nalezení nechtěných artefaktů. Metoda detekce hran není závislá na tvaru zásahu, ale zásahy v terči musí být kontrastní. Hrana je nalezena jen tehdy, je-li rozdíl jasu mezi sousedními pixely dostatečně velký. Jestliže je terč špatně osvícen a vznikne na něm tmavá plocha, ve které je tmavý zásah, hrana nebude nalezena a posléze nebude zásah vyhodnocen. U této metody není třeba velkého výpočetního výkonu, ale nelze jí brát jako finální. Nalezené hrany zásahu mají zpravidla tvar kruhu a z tohoto důvodu použijí v dalším kroku funkci na vyhledání kruhu v obraze Hough Circle Transform.

3.4 Hough Circle Transform

Funkce Hough Circle Transform slouží k nalezení kružnic v obraze a v současné době je v knihovně OpenCV definovaná pouze metoda CV_HOUGH_GRADIENT. Vstupní obraz je nutno před použitím k vyhledání kružnic filtrovat, jinak nedojde k nalezení kružnic.

Základní parametry:

src – vstupní obraz
circles – pole pro uložení souřadnic a poloměru nalezených kružnic
CV_HOUGH_GRADIENT – metoda pro hledání kružnic
dp – inverzní poměr rozlišení
min_dist – minimální vzdálenost mezi detekovanými kružnicemi
param_1 – horní prahová hodnota pro detekci hran
param_2 – práh pro detekci středu
min_radius – minimální velikost kružnice
max_radius – maximální velikost kružnice

Velikost hledaných kružnic je nejlépe nastavit podle ráže, ale musí se zohlednit, že při práci a úpravě obrazu mohlo dojít k malé změně velikosti, a proto je vhodné nastavit parametry s menší rezervou. Vzdálenost mezi středy kružnic je rovná minimální velikosti kružnice, pomocí tohoto omezení by nemělo docházet k nalezení vícero kružnic pro jeden zásah. Horní prahová mez je nastavená tak, aby došlo k nalezení jen zásahů a ne okolního šumu. A práh pro detekci středu se mění v závislosti na počtu nalezených ran.

Ukázka kódu:

Imgproc.HoughCircles(test4, circles, Imgproc.CV_HOUGH_GRADIENT, 1, velikost_stred / 8, 60, presnost, velikost_stred / 12, velikost_stred / 8);

Funkce Hough Circle Transform si nejprve vytvoří matici (x, y), o velikosti původního obrazu. Tato matice má zatím všechny hodnoty nastavené na 0. Je-li poloměr hledané kružnice nastaven dopředu, postačí matice dvourozměrná. V případě, že hledaná kružnice může mít poloměr libovolný, nebo v určitém rozmezí, je třeba vytvořit matici trojrozměrnou (x, y, r). V další fázi se vytvoří z původního obrazu obraz, ve kterém jsou vyznačené hrany podobně jako při použití funkce Canny. Každý nehranový bod je považován za potencionální střed kružnice. Takovýto bod je otestován pro všechny možné velikosti poloměru. Prvek matice, který odpovídá testovanému bodu, se inkrementuje v případě, že je ve vzdálenosti od tohoto bodu nalezen hranový bod a vzdálenost je rovna hledanému poloměru. Prvky matice, které po testování mají největší hodnotu, jsou považovány za středy kružnic o daném poloměru.
Tuto metodu lze brát jako finální segmentaci obrazu, protože dostaneme souřadnice středu kruhu a poloměr kruhu. Metoda CV_HOUGH_GRADIENT není časově náročná a lze jí spustit i vícekrát za sebou s jinými parametry pro dosažení lepšího výsledku.

4 Knihovna OpenCV

4.1 Seznámení se s OpenCV

Knihovnu OpenCV[4] (Open Computer Vision) začala vyvíjet společnost Intel v roce 1999 jako open source knihovnu pro implementaci počítačového vidění, byla publikována jako BSD („Berkeley Software Distribution“)[5] licence, což umožňuje volně šířit licencovaný obsah, a to pouze s uvedením autora a informací o licenci. Knihovna OpenCV zahrnuje práci s počítačovým viděním, a to v reálném čase. Knihovna se využívá pro vytváření aplikací, které rozeznávají objekty, tváře či gesta. Samotná knihovna OpenCV obsahuje stovky algoritmů, které slouží ke kompletní analýze obrazu (např. nástroje pro detekci objektů, pro analýzu pohybu atd.). Výhodou OpenCV je nenáročná komunikace s hardwarem. Aby mohly být použity kamery, není potřeba komunikace s ovladačem hardwaru a nastavování parametrů kamery, neboť je frameworkem komunikováno přímo s daným operačním systémem a je využíváno jeho prostředků. Pokud již tedy máme kameru nainstalovanou v systému, lze ji po inicializaci hned použít. Knihovna OpenCV může být využita i pro jazyky C, C++ či Python. Primárně je knihovna OpenCV určena pro platformy Windows, Linux a Mac, ale nyní je už hojně využívána i na mobilní platformy Android a iOS.

4.2 Popis základních funkcí OpenCV

4.2.1 Třída MAT

Základním prvkem pro práci s obrazy v OpenCV je třída Mat. Při vytváření třídy Mat je vždy využito tolik paměti, kolik je zrovna potřeba a programátor nemusí řešit alokaci a uvolňování paměti. Třída Mat má dvě datové části, první je velikost matrice X Y a druhá část je typ metody používané pro uchování dat.

Základní parametry: rows - počet řádků v 2D poli
cols – počet sloupců v 2D poli
type – barevný model

V programu je nutné vytvořit instanci třídy Mat, která reprezentuje obraz. Tento obraz je poté možné buď načíst, nebo vytvořit. Uložený obraz je tvořen 2D polem hodnot. Například 8bitový barevný obraz RGB má každý svůj pixel uložený jako strukturu tří parametrů, jeden parametr na jednu barevnou složku a každý parametr dosahuje hodnot od 0 do 255. Takže čistě zelený bod bude mít hodnotu [0,255,0]. Černobílý obraz (obraz ve stupni šedi) je tvořen jen jedním parametrem.

Ukázka kódu načtení obrázku:

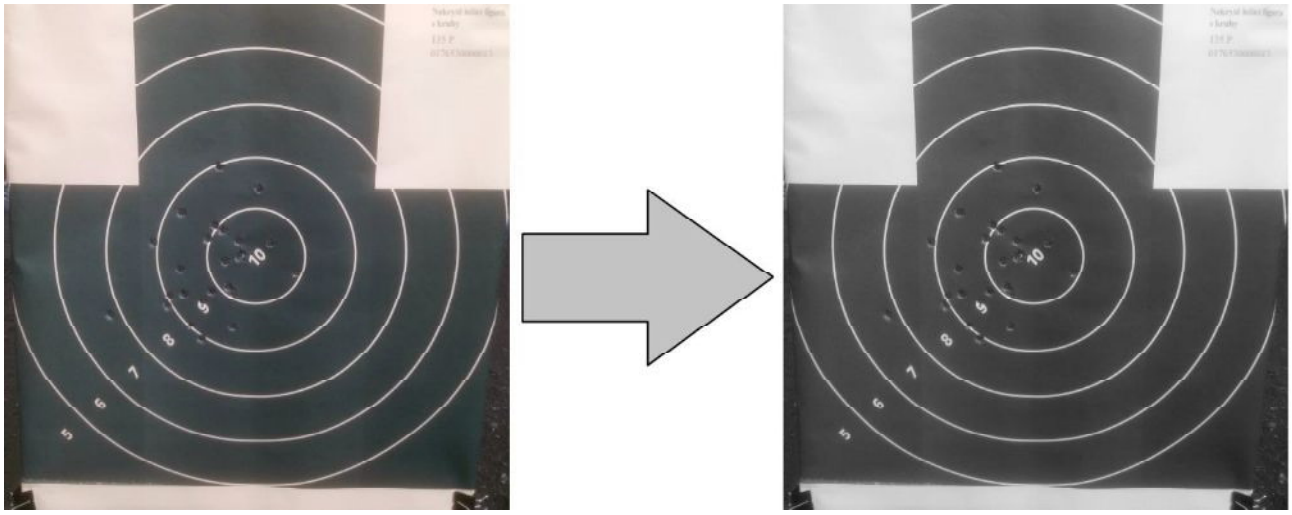
```
Mat test =  Imgcodecs.imread(cesta);
```

[4] OpenCV. [online]. [cit. 2016-03-17]. Dostupné z: http://opencv.org/
[5] BSD. [online]. [cit. 2016-03-17]. Dostupné z: https://cs.wikipedia.org/wiki/BSD

4.2.2 Funkce cvtColor

Funkce cvtColor slouží k převodu barevných modelů. Lze pomocí této funkce převést barevný obrázek na černobílý, který je vhodnější pro analýzu. Při převodu RGB2GRAY dochází k vynásobení každé barevné složky určeným poměrem a následným sečtením do jedné hodnoty.
Hodnota šedé = R(červená) * 0.299 + G(zelená)*0.587 + B(modrá)*0.114
Základní parametry: src – vstupní obraz
dst – výstupní obraz
code – typ převodu (COLOR_BGR2RGB , COLOR_RGB2GRAY)
DstCn – počet kanálů ve výstupním obrazu
Ukázka kódu převodů z barevného na černobílý :

Imgproc.cvtColor(test1, test2, Imgproc.COLOR_RGB2GRAY, 4);

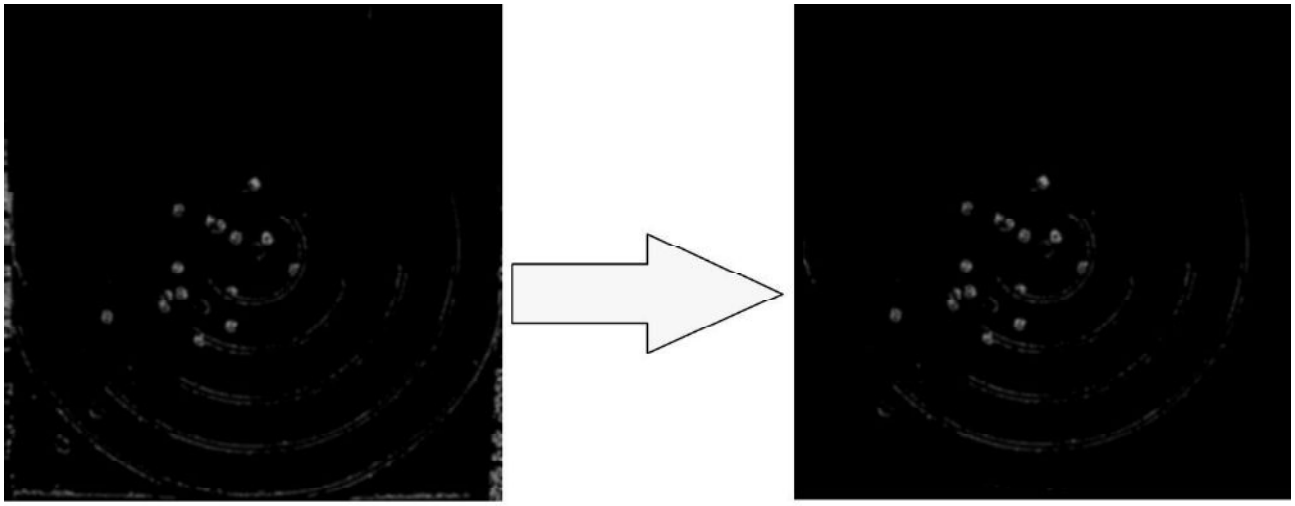


Ilustrace 2: funkce cvtColor, převod barevného modelu na černobílý

4.2.3 Funkce setTo

Funkce setTo je využívána k nastavení celého obrazu na určitou barvu. Lze u této metody použit parametr mask, který slouží jako maska, podle které se změní jen určitá část obrazu. Tuto metodu jsem použil při zakrytí terče a překreslení okolí na černou barvu.
Ukázka kódu :

test3.setTo(new Scalar(0, 0, 0), mask);

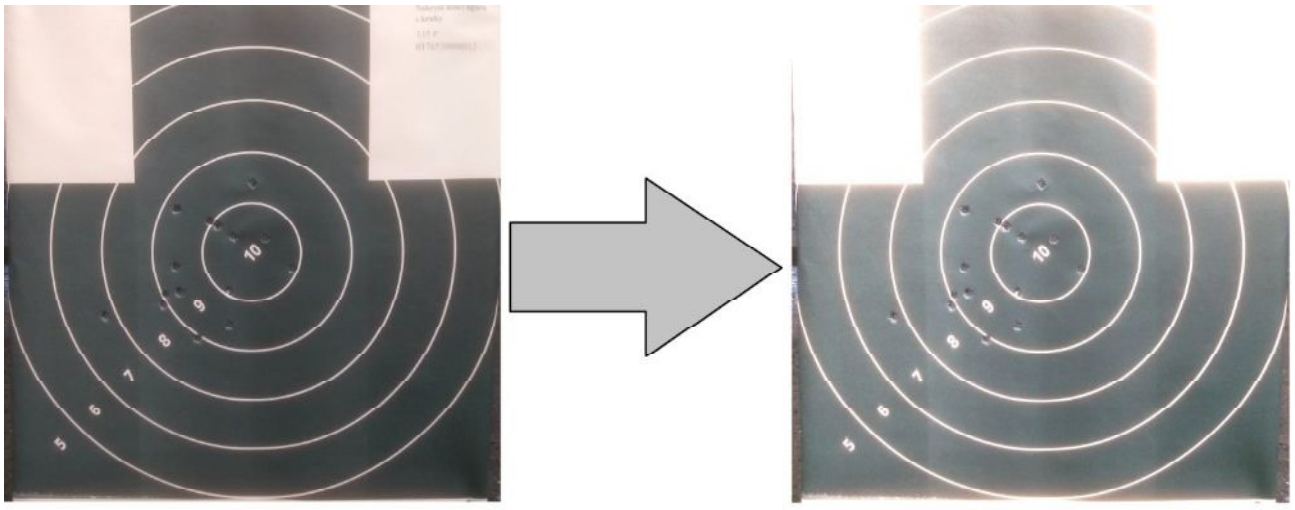


Ilustrace 3: funkce setTo, zakrytí části obrazu pomocí masky

4.2.4 Funkce convertTo

Funkce convertTo má čtyři parametry. První výstup, do kterého chceme výsledný obraz uložit, druhý je počet kanálů výstupního obrazu a při použití záporného parametru zůstane počet stejný jako u vstupního obrazu. Třetí a čtvrtý parametr je nejdůležitější. Třetí parametr je desetinné číslo, kterým se každý pixel v obraze vynásobí a tím zvýší svůj jas. Čtvrtý parametr je celé číslo, které se přičte ke každému pixelu. Já tuto metodu použil, když jsem potřeboval světlé body ještě více zvýraznit a tmavé body odstranit.
Ukázka kódu:

test4.convertTo(test4, -1, 1.8, 20);



Ilustrace 4: funkce convertTo, zvýšení jasu

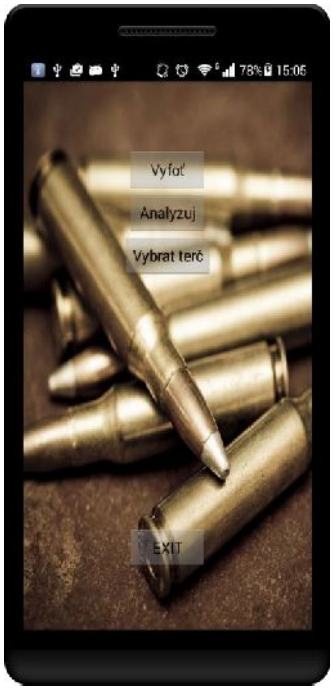
5 Aplikace analyzátor

5.1 Menu – popis jednotlivých možností

Menu celé aplikace je velmi jednoduché a snadné na ovládání. Velký důraz byl kladen na to, aby uživateli stačilo jen pár kliknutí k výsledné analýze jeho terče. Na úvodní stránce, která slouží jako hlavní menu, jsou celkem čtyři tlačítka.

- Vyfoť - slouží k vyfocení nového terče
- Analyzuj - spustí první krok analýzy na posledním vyfoceném terči
- Vyber terč - uživatel může zvolit jiný už vyfocený terč, který má v paměti zařízení
- EXIT – ukončí aplikaci

V průběhu dalšího vývoje bude aplikace vylepšená o další tlačítko, a to tlačítko Nastavení (viz kapitola 8.)



Ilustrace 5: hlavní menu

5.2 Vyfocení terče

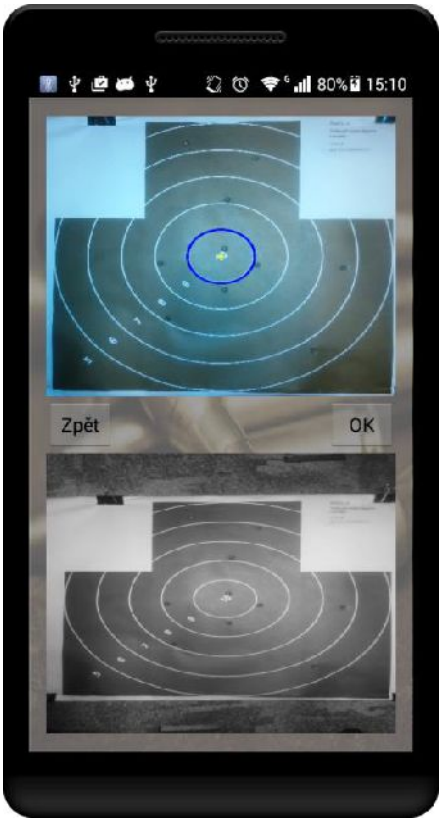
Vyfocení terče, stejně jako celá aplikace, je uživatelsky velice snadné. Aplikace si sama zjistí všechna možná rozlišení, která fotoaparát na daném zařízení zvládá. Následně si vybere z listů možností to, které nejvíc vyhovuje pro analýzu. Uživatel jen musí zamířit středové kolečko, které vidí na displeji,

na střed svého terče, a dále pomocí zeleného rámečku, který je vykreslen také na displeji, srovnat mobil tak, aby v něm byl celý terč. Aplikace sice umí nalézt střed terče, a sama si terč oříznout, ale při špatném úhlu focení dochází k nepoměru stran. Má-li uživatel mobil správně namířený na terč, stačí jen kliknout na tlačítko Vyfotit v pravém dolním rohu a aplikace provede automatické ostření, terč vyfotí a uloží. Lze pořídit více obrazů terčů a poté je až analyzovat.

Ilustrace 6: focení terče

5.3 Analýza terče

Analýza terče spočívá ve dvou krocích. V prvním kroku dochází ke kontrole, zda se vůbec jedná o terč a zda jsou všechny zásahy dobře čitelné. Uživateli je výsledek zobrazen na displeji a ten se může následně rozhodnout, jestli v analýze pokračovat tlačítkem OK nebo dát tlačítko Zpět a vyfotit terč znovu. Kontrola terče funguje tak, že aplikace hledá kruh v oblasti středu obrázků, a to o velikosti, který by měl odpovídat desetibodovému kruhu na terči. Není-li takový kruh nalezen, není možné dál pokračovat v analýze. Čitelnost zásahu provede sám uživatel, kterému je zobrazen terč černobíle, a zásahy by měly být viditelné na první pohled. V horním okně se zobrazuje uživateli nalezení středu terče a ořez terče. Ve spodním okně se zobrazuje terč černobíle.



Ilustrace 7: první krok analýzy

V druhém kroku probíhá analýza terče a zásahů v něm. Ze zjištěného středu a velikosti desetibodového kruhu na terči lze ořezat přesně terč a určit zbývající bodové kruhy podle vzdálenosti od středu. Obraz terče se upraví tak, abychom dostali černé plátno a v něm bílé pixely, které odpovídají zásahům. Na takto upravený obraz spustíme funkci pro vyhledání kruhů, která má nastavené parametry podle zadané ráže střeliva uživatelem (zatím 9 mm). Funkce vrátí pozici nalezených kruhů a podle vzdálenosti od středu terče aplikace vyhodnotí, o jaký bodový zásah se jedná. V horním okně je uživateli zobrazen terč s barevně vyznačenými zásahy a v dolním okně je soupis a celkový počet zásahů a výsledné bodové skóre. Uživatel může výsledek uložit stisknutím tlačítka Uložit, nebo vylepšit analýzu terče stisknutím tlačítka Vylepšit, a poté se upraví přesnost vyhledávaní kružnic v závislosti na poměru počtu nalezených a uživatelem zadaných ran.

Ilustrace 8: výsledná analýza a vyhodnocení

6 Algoritmus krok za krokem

6.1 Načtení a příprava terče

Aplikace si zvolený terč načte do třídy Mat. Knihovna OpenCv načítá terč v modelu BGR, což může být trochu matoucí, a proto si ho pomocí funkce cvtColor popisované v předešlé kapitole (3.) převedu na RGB. Dále si pomocí stejné funkce vytvořím kopii obrazu, ale ve stupních šedé barvy. Barevný terč ve formátu RGB zesvětlím pomocí metody converTo, čímž dostanu výraznější přechody mezi světlou a tmavou barvou. Abych se zbavil šumu v obraze, který vznikne při focení, aplikuji na něj funkci blur. Používal jsem také funkci Gaussian Blur, která obraz vyfiltruje lépe, ale trvá déle než blur a konečné výsledky v analýze jsou srovnatelné.

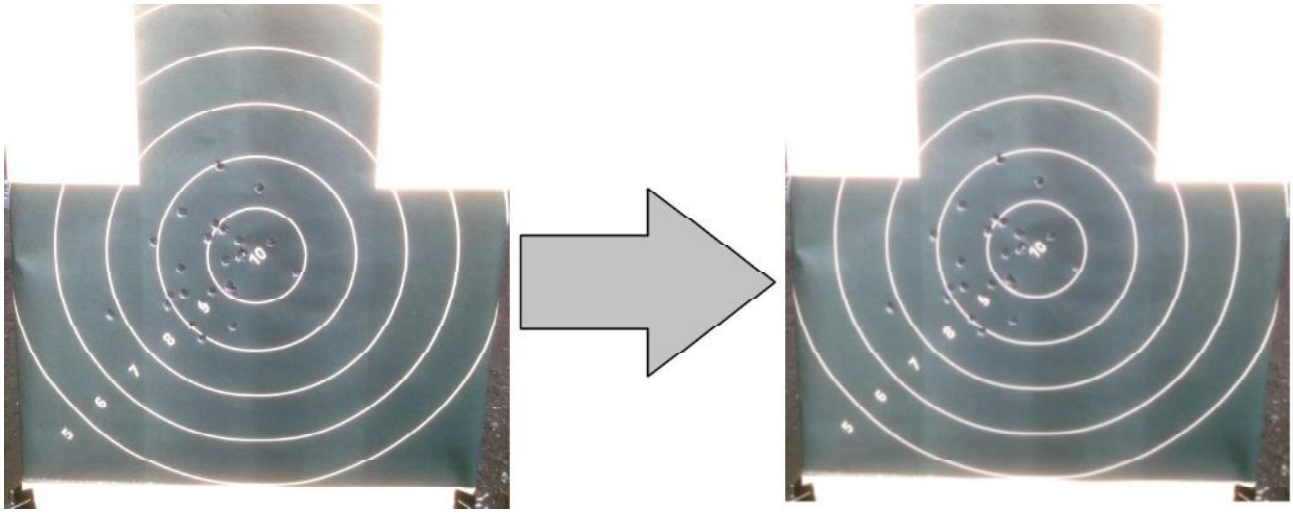
6.1.1 Blur

Blur je základní filtrovací funkce. Funkce blur vypočítává hodnotu každého pixelu na základě hodnoty jádra a hodnoty sousedních pixelů z původního obrazu. Důležité je nastavit správnou velikost jádra, je-li velikost jádra příliš velká, může dojít ke ztrátě drobných prvků v obraze a je-li velikost příliš malá, nemusí být dostatečně odstraněn šum.

Základní parametry: src – vstupní obraz
dst – výstupní obraz
ksize – velikost jádra
anchor – výchozí bod v jádru, záporné číslo znamená střed jádra

Ukázka kódu:

Imgproc.blur(img2, img2, new Size(9, 9), new Point(-1,-1));



Ilustrace 9: filtrovací funkce blur

6.2 Získání masky z terče

Maska je v analýze obrazu něco jako šablona, podle které aplikujeme na jiný obraz nějakou barvu. Můžeme pomocí masky danou část obrazu zakrýt nebo naopak nechat viditelnou jen určitou část. V mém případě musím získat z terče bodové kruhy a čísla udávající body daného kruhu. Využiji toho, že jsou bílé barvy a tudíž to nejsvětlejší místo na celém terči, a pomocí funkce inRange a vhodně nastavených parametrů je vytáhnu z obrazu.

6.2.1 InRange

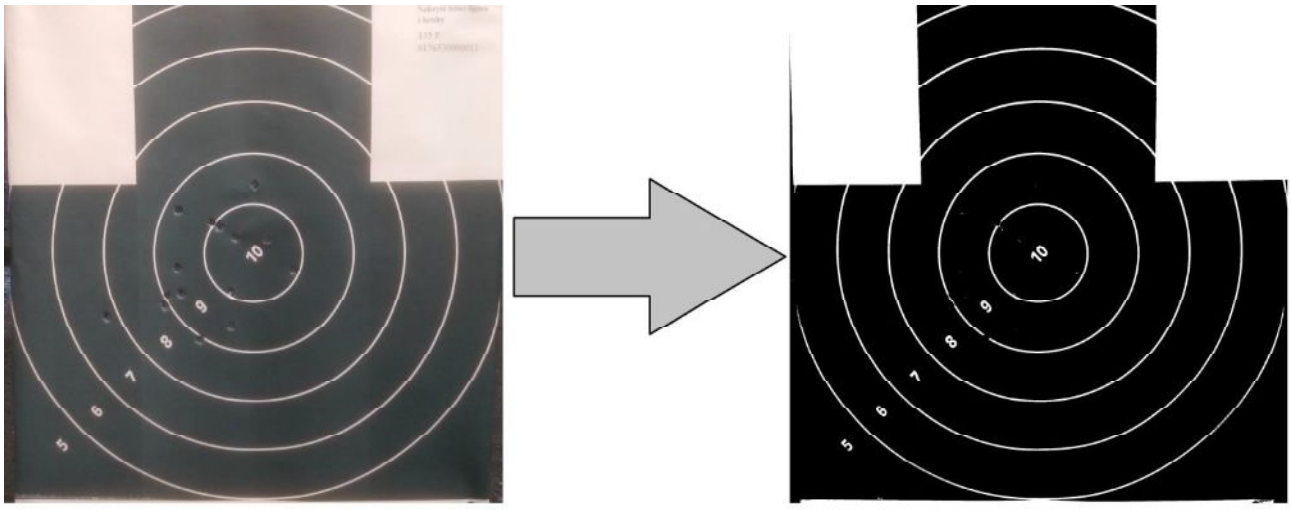
InRange je funkce, která prochází každý pixel v obraze a kontroluje, zda hodnota daného pixelu je v rozmezí barev, které jsme zadali. Je-li hodnota v rozmezí, tak do výstupního obrazu se na stejné pozici zaznamená bílý pixel, v opačném případě pixel černý.

Základní parametry: src – vstupní obraz
lowerb – nejmenší hodnota barev v rozmezí
upperb – největší hodnota barev v rozmezí
dst – výstupní obraz

Funkce inRange lze aplikovat jak na barevný obraz, kde musíme zadat hodnotu všech tří barev, tak na obraz ve stupních šedé. Já tuto funkci zkoušel použít na oba druhy obrazů, ale při použití terče ve stupních šedé jsem dosahoval lepšího výsledku.

Ukázka kódu:

Core.inRange(testGray, new Scalar(170, 170, 170), new Scalar(255, 255, 255), mask);



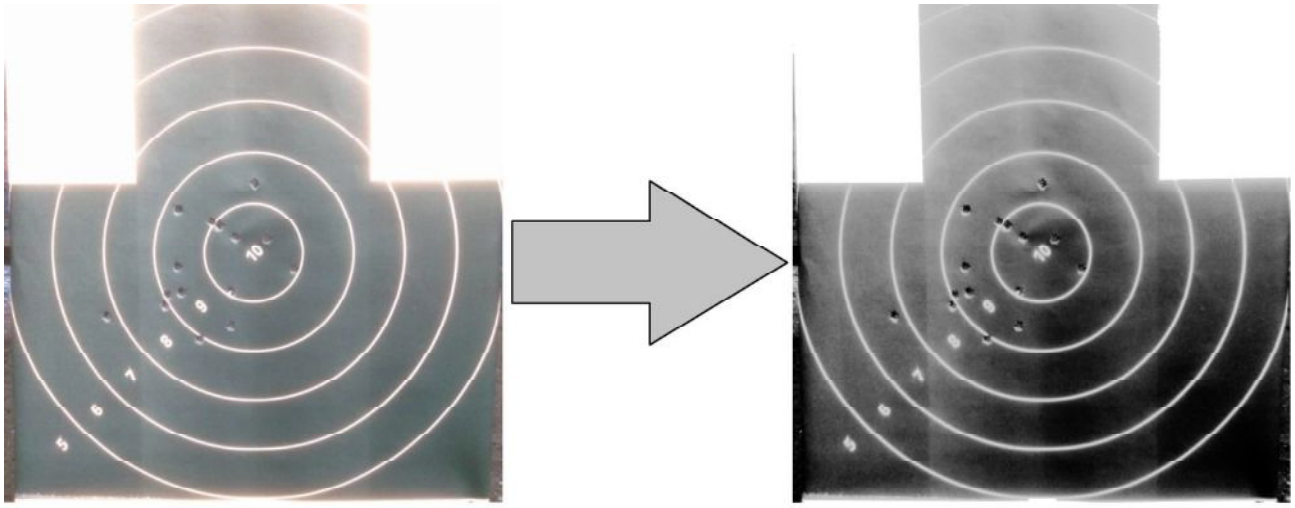
Ilustrace 10: funkce inRange, výběr bílé barvy

6.3 Zvýraznění zásahů

Po aplikování funkce blur na barevný obraz terče nejsou zásahy stále dobře viditelné. Zásahy musí být dobře čitelné a zvýrazněné, aby se eliminovala možnost chyb. V ideálním případě se snažím docílit bílých kruhů na černém pozadí, kde každý bílý kruh bude pozice a velikost zásahu. Obraz je ale vyčištěný od šumu a lze na něj provést funkci equalizeHist, a poté funkci Canny na detekci hran.

6.3.1 EqualizeHist

Funkce equalizeHist vyrovnává rozložení intenzity v obraze. Histogram je jakási statistika jednotlivých pixelu a jejích jasů. Histogram u 8bitového obrazu pracuje s hodnotami 0 až 255, kde nula je černá a 255 bílá. Většina obrazových bodů na terči má hodnotu někde uprostřed, tudíž kolem hodnoty 126. Funkce equalizeHist rozšíří a rovnoměrněji rozdělí hodnoty intenzity do celého rozsahu. Parametry jsou vstupní a výstupní obraz.



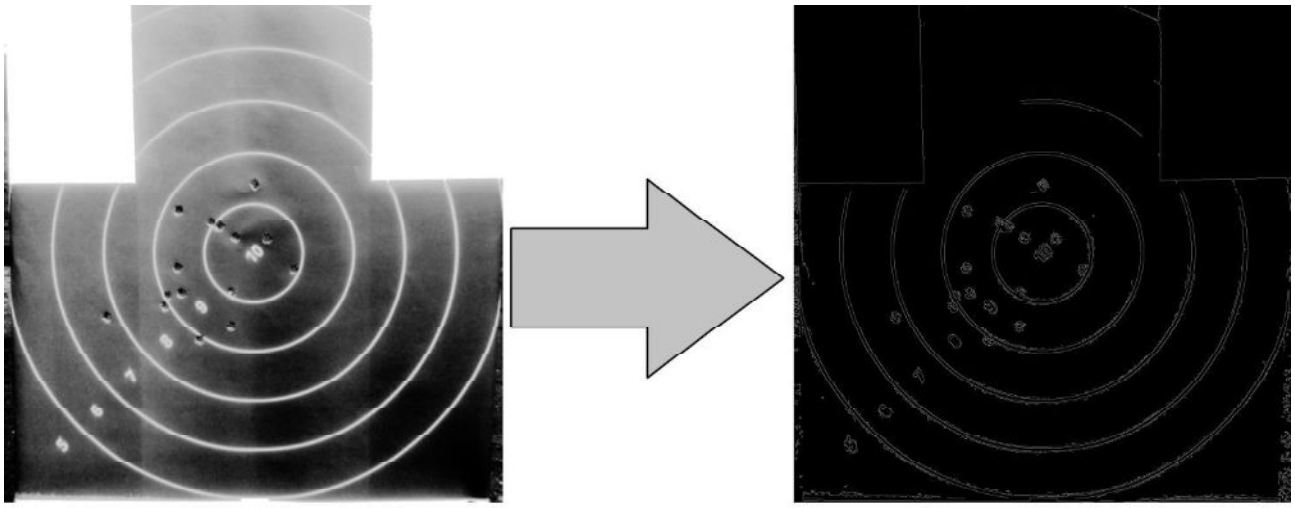
Ilustrace 11: funkce equalizeHist, zvýraznění zásahu

6.3.2 Canny

Canny je funkce pro detekci hran v obraze, kterou vymyslel Johny F. Canny v roce 1986. Funkce hledá v obraze rozdíly intenzit sousedních pixelů, a vrací obraz, ve kterém jsou bílou barvou znázorněny hrany na černém pozadí.

Základní parametry: src – vstupní obraz ve stupních šedi
dst – výstupní obraz
lowThreshold – dolní prahová hodnota
highThreshold – horní prahová hodnota
kernel_size – velikost jádra

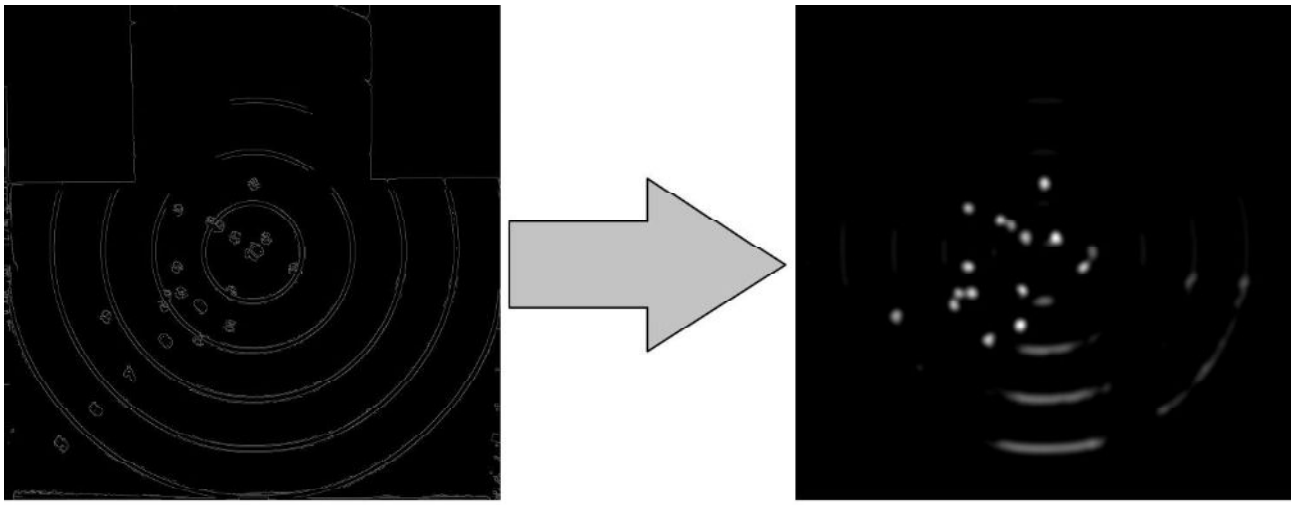
Hrana je nalezena, jestliže rozdíl intenzit sousedních pixelů je vyšší než horní prahová hodnota. V případě, že rozdíl hodnoty intenzit je v rozmezí mezí dolní a horní prahovou hodnotou, bude hrana nalezena jen v případě, bude-li napojena na již nalezenou hranu.



Ilustrace 12: funkce canny, detekce hran

6.4 Odstranění kruhů a zvýraznění zásahů

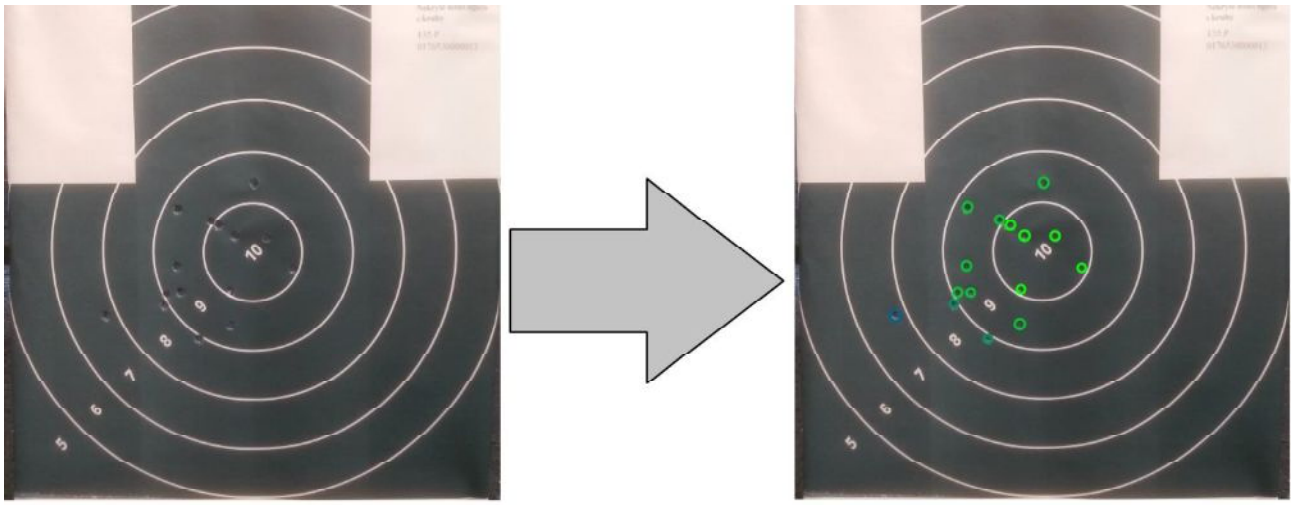
Zásahy po detekci hran jsou již dobře viditelné, ale z důvodu, že v obraze jsou vyznačené i bodové kruhy a čísla, mohlo by docházet k nalezení i jiných kruhů, které by neodpovídaly pouze zásahům. Proto musím bodové kruhy odstranit nebo aspoň potlačit, a tak zvýraznit zásahy natolik, aby nedocházelo k chybné identifikaci. Proto obraz s hranami překryji, pomocí již vytvořené masky, kterou jsem pomoci funkce blur trochu rozmazal, černou barvou a to pomocí funkce setTo. Překrytí mi ale neodstraní všechny nedostatky v obraze, a tak využijí funkce blur, která celý obraz rozmaže a drobné nedostatky odstraní.



Ilustrace 13: odstranění kruhů

6.5 Nalezení a vyznačení zásahu

Obraz terče už mám dostatečně upravený, abych v něm mohl hledat kružnice, které budou znázorňovat zásahy. K nalezení kružnic použijí Hough Circle Transform. Kružnice zvýrazním do původního obrazu terče a podle vzdálenosti od středu, který jsem našel při prvním kroku analýzy, vypočítám jejich bodové hodnocení. Musím vzít v potaz ráži, kterou byly zásahy udělané, a hledat pouze kruhy správné velikosti.



Ilustrace 14: funkce Hough Circle Transform, vyhodnocení zásahů

7 Testování

Testování je důležitou součástí vývoje jakéhokoliv softwaru. Já svůj algoritmus otestuji jak z hlediska rychlosti, tak přesnosti nalezení zásahů. Aplikaci budu testovat na rozlišných zařízeních a použiji k testům rozdílně vyfocené terče za rozdílných světelných podmínek. Všechny terče budou prostřelené stejnou ráží a to ráží 9 mm luger, avšak s použitím rozdílných typů střel, které v terči zanechají jiný tvar zásahu. První test bude test rychlosti analýzy.

7.1 Rychlost analýzy

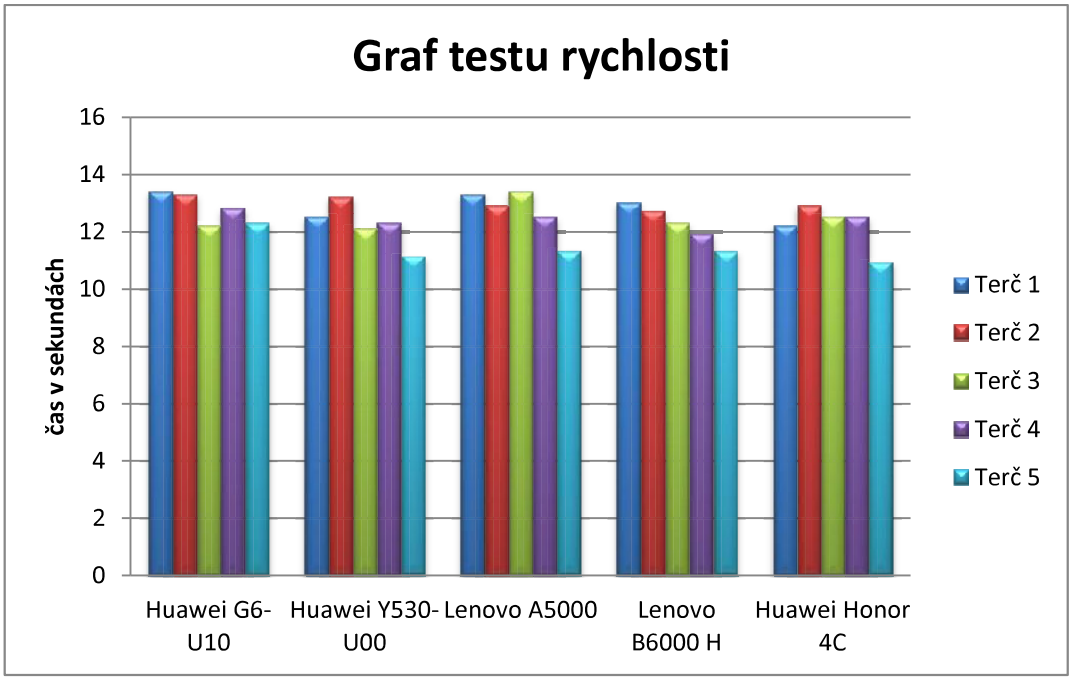
K testu rychlosti použiji 5 druhů mobilních zařízení s jiným hardwarem a operačním systémem.

1. mobilní telefon Huawei G6-U10	**CPU: 1190Mhz x 4 RAM: 912MB**	**OS: Android 4.3**
2. mobilní telefon Huawei Y530-U00	**CPU: 1190Mhz x 2 RAM: 410MB**	**OS: Android 4.3**
3. mobilní telefon Lenovo A5000	**CPU: 1300Mhz x 4 RAM: 956MB**	**OS: Android 5.0.2**
4. tablet Lenovo B6000 H	**CPU: 1209Mhz x 4 RAM: 968MB**	**OS: Android 4.4.2**
5. mobilní telefon Huawei Honor 4C	**CPU: 1200Mhz x 8 RAM: 2048MB**	**OS: Android 5.1.1**

Na těchto zařízeních budu testovat sérií 5 terčů vyfocených za různých světelných podmínek. Terče jsou focené na 8MegaPixelový fotoaparát s rozlišením 2448 X 3264 s velikostí od 600KB do 1MB. Čas budu měřit u druhého kroku analýzy, kdy dochází k provedení celého algoritmu. Z testů očekávám, že na výkonnějším zařízení, které má větší paměť RAM a více jader procesoru, bude analýza trvat kratší dobu

Test rychlosti (s)	Terč 1	Terč 2	Terč 3	Terč 4	Terč 5
Huawei G6-U10	13,4	13,3	12,2	12,8	12,3
Huawei Y530-U00	12,5	13,2	12,1	12,3	11,1
Lenovo A5000	13,3	12,9	13,4	12,5	11,3
Lenovo B6000 H	13	12,7	12,3	11,9	11,3
Huawei Honor 4C	12,2	12,9	12,5	12,5	10,9

Tabulka 1: test rychlosti



Ilustrace 15: graf testu rychlosti

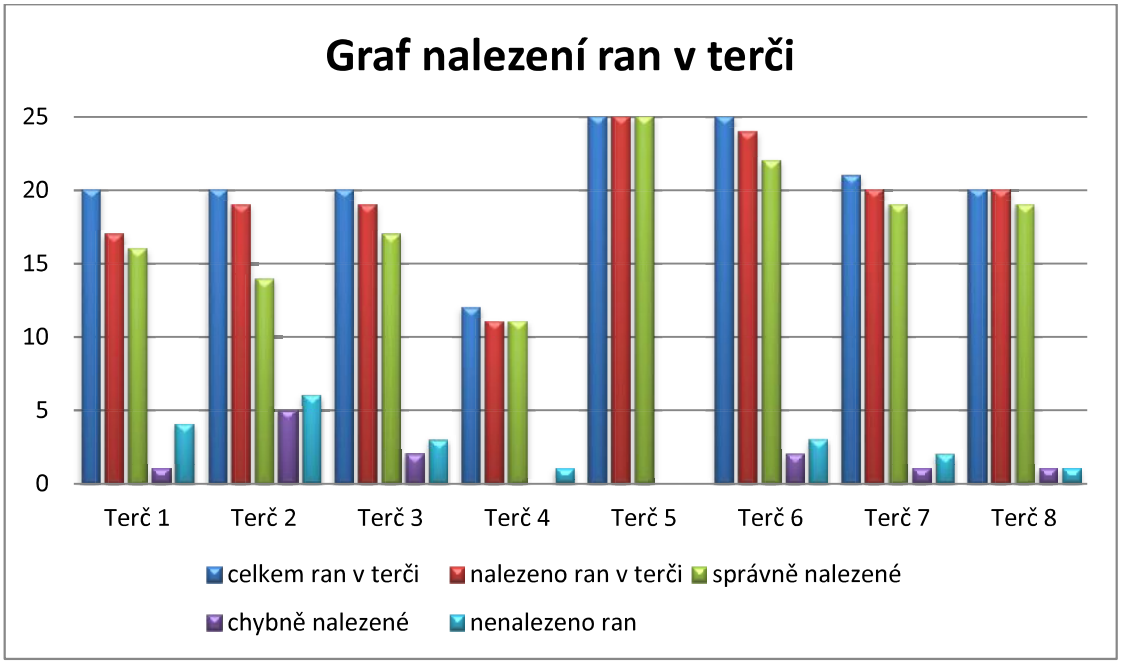
Z výsledku testů jde vidět, že počet jader procesoru, paměť RAM ani operační systém nemají na rychlost analýzy vliv. Při zkoumání důvodu proč tomu tak je, jsem zjistil, že knihovna OpenCV zvládne využít pouze dvě jádra procesoru. Průměrný čas analýzy se pohyboval okolo 12 vteřin, což je pro uživatele přijatelné.

7.2 Přesnost analýzy

Možná tím nejdůležitějším faktorem analýzy je její přesnost. U analýzy obrazu a hledaní objektu v něm musíme vždy počítat s malým procentem chybného nalezení. Já otestuji svůj analyzátor na sériích několika obrazů terčů. Stejně jako v předchozím testu použijí terče vyfocené za jiných světelných podmínek a navíc zkusím použít obrazy, které jsou vyfocené na jiném mobilním zařízení s jiným fotoaparátem. Terč 1 až 5 má rozlišení 2448x3264 a terč 6 až 8 má rozlišení 1944x2592. Věřím, že při testu se projeví přesnost nalezení zásahu v závislosti na kvalitě obrazu vyfoceného terče. Tento test není potřeba testovat na různých zařízeních, protože výsledek by byl pokaždé stejný. V prvním testu se zaměřím na celkový počet nalezených zásahů, počet chybně nalezených zásahů a počet správně nenalezených zásahů. V druhém testu se zaměřím na počet bodů, které analyzátor nalezne, a ty porovná s bodovým hodnocením, které provede certifikovaný rozhodčí.

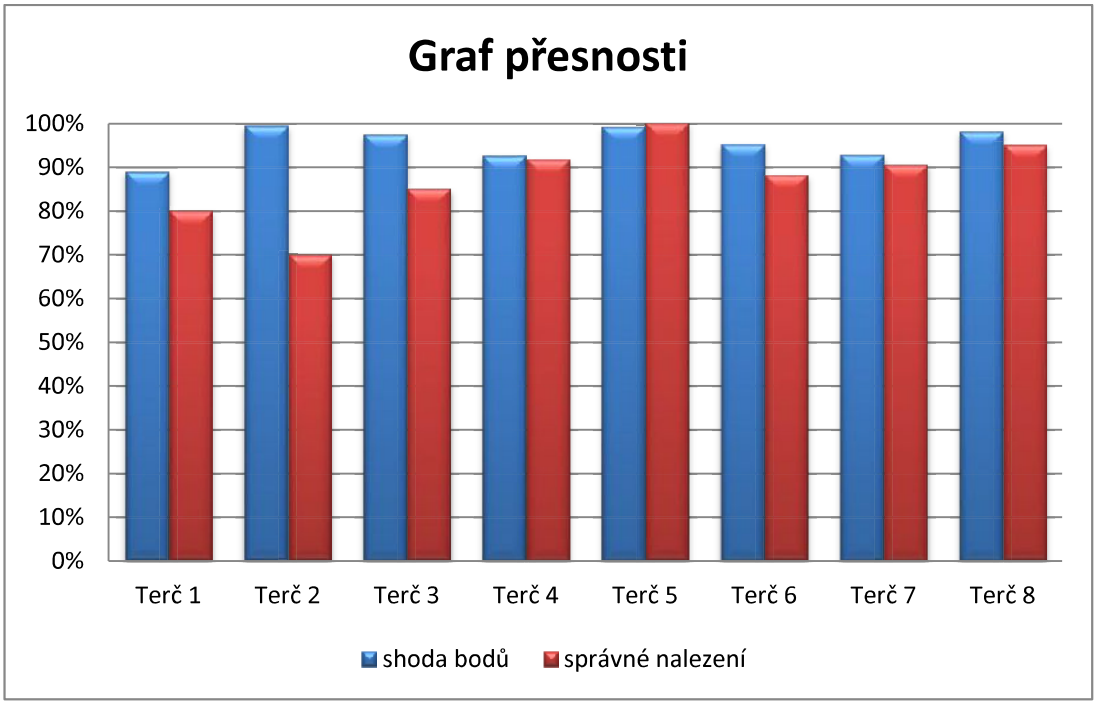
Test přesnosti	Terč 1	Terč 2	Terč 3	Terč 4	Terč 5	Terč 6	Terč 7	Terč 8	Celkem
celkem ran v terči	20	20	20	12	25	25	21	20	
nalezeno ran v terči	17	19	19	11	25	24	20	20	
správně nalezené	16	14	17	11	25	22	19	19	
chybně nalezené	1	5	2	0	0	2	1	1	
nenalezeno ran	4	6	3	1	0	3	2	1	
body (rozhodčí)	163	161	161	95	228	228	179	161	
body (analyzátor)	145	162	157	88	226	217	166	164	
shoda bodů	89%	99%	98%	93%	99%	95%	93%	98%	**95,46%**
správné nalezení	80%	70%	85%	92%	100%	88%	90%	95%	**87,52%**

Tabulka 2: test přesnosti



Ilustrace 16: graf nalezení ran v terči

Jak z grafu vyplývá, tak dochází při analýze k nalezení podobného počtu ran, kolik je v terči zásahů. Ale ne vždy jsou tyto zásahy správně ohodnoceny nebo vyznačeny, proto se přesnost shody bodů pohybuje okolo 95 %, zatímco přesnost nalezení zásahu jen okolo 88 %. Pro lepší funkčnost analýzy je důležitější správnost nalezení zásahu než shoda bodů, u které může být velkým faktorem náhoda.



Ilustrace 17: graf přesnosti

Z výsledků testování lze říci, že i 5Mpix fotoaparát na mobilním zařízení má dostatečnou kvalitu na analýzu terčů. U obrazů nižší kvality byla při testech zpozorována vyšší rychlost analýzy. Přesnost analýzy byla překvapivě vysoká, a to okolo 88 %. Ale, jak již bylo zmíněno výše, toto vysoké číslo můžou ovlivnit jak světelné podmínky, tak sám uživatel, který nevyfotí terč úplně správně.

8 Možné vylepšení aplikace

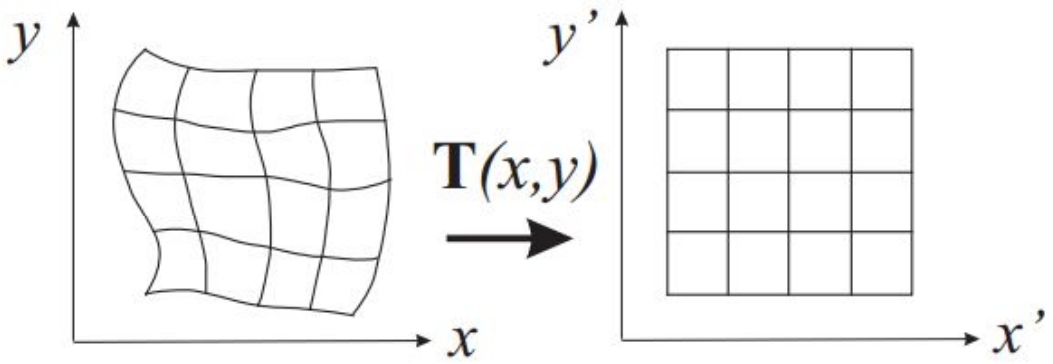
Tato kapitola bude zaměřena na různé modifikace a vylepšení aplikace pro analýzu terčů. Při vývoji a následném testování jsem objevil pár nedostatků, které by bylo možné odstranit. Bohužel některá tato vylepšení budou náročnější na výpočet a budou tudíž trvat delší dobu, což by bylo uživatelsky nepřijatelné.

8.1 Ořezávání terče

Aplikace má již v sobě zabudované ořezání terče, které slouží pro přesnější a rychlejší výpočet, jelikož nemusí být počítáno s takovým množstvím dat, avšak ořezávání je závislé na správném úhlu vyfocení terče. Je-li terč vyfocen pod špatným úhlem, tak dochází k nepoměru protilehlých stran, což způsobuje, že ořezávání podle středu a jeho velikosti a předem naprogramované šablony nelze použít. Jedna z možností, jak takový terč oříznout, je nalezení kontur terče pomocí hran. I v takovém případě bude terč stále zkosený, a proto by bylo vhodnější jej transformovat.

8.2 Transformace terče

Transformace slouží k pootočení a k vyrovnání poměru stran. Pokud je terč vyfocen tak, že je jedna strana menší než protilehlá, dochází ke špatnému vyhodnocení ran. Při transformaci se může část obrazu nacházet mimo výsledný obraz, ale aplikace je udělána tak, aby terč měl dostatek místa kolem sebe.



Ilustrace 18: pootočení a vyrovnání poměru stran

8.3 Zvýšení přesnosti nalezení zásahu

Jak bylo zjištěno při testování přesnosti, aplikace v této chvíli nevyhledává zásahy zcela přesně. Tato situace by šla vyřešit jemnějším rozostřením a použitím složitější metody, což by mělo za následek mnohonásobně delší dobu analýzy. Vývojem technologie by mohl být v mobilních zařízeních hardware výkonnější, tudíž by byl výpočet rychlejší, a šlo by použít tedy i složitější metody. V testech bylo zjištěno, že s lepšími parametry fotoaparátu bylo vyhledávání přesnější.

8.4 Analýza terče při focení

Dalším možným vylepšením by bylo základní analýzu terče provést již při focení. Aplikace by rozeznala terč, jeho střed a obrys, již před analýzou, tudíž by uživatel věděl, zda vyfocení terče proběhlo správně. Tohle by bylo velice náročné na výpočet při snímání terče a musela by se omezit snímková frekvence zobrazení.

8.5 Uživatelské rozhraní

Aplikace bude časem rozšířena o nastavení možností uživatelem, který si bude moct nastavit typ terče, ráži, kterou střílel a počet vystřelených ran do terče. Všechna tato nastavení budou mít vliv pro analýzu a bude ji to zpřesňovat. Dále by se dalo nastavit i počasí a světelné podmínky, které by také zlepšovaly analýzu, ale bylo by potřeba provést spoustu testů v různých podmínkách a vyhodnotit je.
Tím že analýze není přesná na 100%, je nutné dodělat uživatelské rozhraní o možnost manuálního přidání a odebraní zásahu.

9 Závěr

Cílem mé bakalářské práce bylo seznámení se s metodami pro analýzu objektů v obraze, vybrat vhodné metody pro nalezení zásahu z vyfoceného terče a pomocí těchto metod naimplementovat detektor. S vývojem mobilních zařízení a jejích přístupnosti jsem se rozhodl tento detektor naimplementovat jako mobilní aplikaci.

V prvním kroku bylo nezbytné se seznámit s metodami pro analýzu objektů v obraze. Nebylo by časově reálné vyzkoušet každou známou metodu pro analýzu objektů v obraze, a proto bylo nutné, abych se na samém začátku zamyslel a vytřídil jen ty metody, které měly šanci na úspěšné detekování zásahu v terči. Vybrané metody jsem posléze vyzkoušel a popsal, jakých výsledků jsem dosáhl. Knihovnu OpenCV, která slouží pro práci s obrazem a má už spoustu metod v sobě, jsem před tím nikdy nepoužil, a tak bylo potřeba naučit se její základní třídy a funkce. V dnešní době lze použít knihovnu OpenCV i na operační systém Android, který je napsán v jazyce Java. Bohužel je to poměrně nová záležitost a dokumentace a jiné informace nejsou tak obsáhlé, jako pro jazyk C++.

Při tvorbě aplikace byl kladen důraz na jednoduché a rychlé ovládání. Uživatel, který používá aplikaci na analýzu terčů, by měl být schopný během krátké doby vyfotit a nechat si zanalyzovat terč a následně uložit.

V průběhu vývoje algoritmu pro detekci zásahu jsem zjistil, že velký vliv na správnou detekci zásahu má správné vyfocení terče. Vymyslel jsem proto rámeček, který přesně orámuje focený terč a podle kterého ho může uživatel vyfotit. Aby nedocházelo ke zbytečné analýze chybně vyfocených terčů, je ještě před samotnou analýzou ověřeno, zda se jedná opravdu o terč, který bude možno analyzovat. Takovýto terč je poté předpřipraven pro metodu, která v něm vyhledá kruhy značící zásahy.

Testy analýzy jsem rozdělil do dvou kroků, a to na test rychlosti a test přesnosti. U testu rychlosti mě výsledky trochu zklamaly, protože doba analýzy trvala při použití 8Mpx obrázku kolem 12 vteřin i přes mé snahy udělat algoritmus co nejrychlejší. Naopak při testu přesnosti se ukázalo, že shoda bodů vypočítaných pomoci aplikace a pomocí rozhodčího je kolem 95 %. A šance na správné nalezení zásahu je skoro 88 %.

Analýza je velice citlivá na světelné podmínky, při kterých je terč vyfocen. V poslední kapitole jsem navrhl pár možných vylepšení, která by mohla tuhle citlivost snížit a aplikaci vylepšit.

Vývoj aplikace prověřil a zdokonalil mé znalosti z jazyka Java. Blíže jsem se seznámil s operačním systémem Android a naučil jsem se využívat periférie mobilních zařízení jako například kameru. Při průzkumu metod na rozpoznávání objektů v obraze a později při implementaci jsem se dozvěděl spoustu informací jak pracovat s obrazem.

10 Reference

[1] LEX. [online]. [cit. 2016-03-18]. Dostupné z: http://gunlex.cz/clanky/informace-lex/2002-pocet-drzitelu-zbrojnich-prukazu-a-zbrani-v-roce-2014

[2] M. Šonka, V. Hlaváč: Počítačové vidění, Grada. Praha 1992, ISBN 80–85424–67–3.

[3] M. Dobeš: Zpracování obrazu a algoritmy v C#, BEN – technická literatura. Praha 2008–233–6, ISBN 978–80–7300

[4] OpenCV. [online]. [cit. 2016-03-17]. Dostupné z: http://opencv.org/

[5] BSD. [online]. [cit. 2016-03-17]. Dostupné z: https://cs.wikipedia.org/wiki/BSD

Příloha na CD

Na přiložené CD jsou následující soubory:

1. Text bakalářské práce ve formátu PDF
2. Aplikace analyzátor pro nainstalování na Android
3. Složka (Terče) obsahující 8 testovaných terčů